

# **Pipeline Proposal**

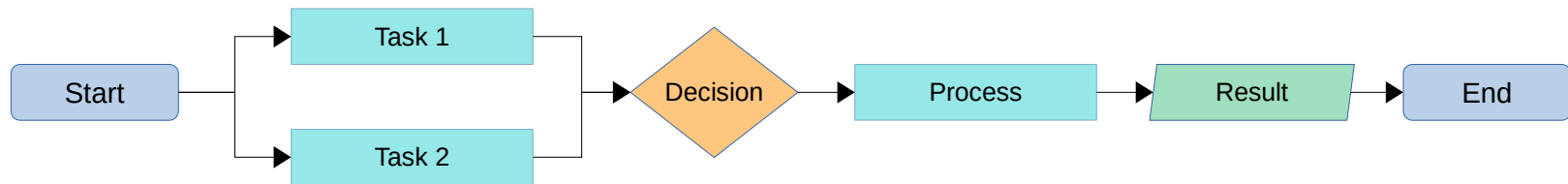
1. Introduction
2. User Discipline
3. Project
4. Category
5. Step or Task
6. Versions
7. Version Components
8. Configure the Pipeline Package and Project
9. Let's start the works
10. Work flow
11. Detailed Work flow with specific step
12. Folder Structure
13. Versioning
14. Step Published Components (Data Types)
15. Core Pipeline Production Tools
16. Pipeline Frameworks and Packages (Repository)

## 1. Introduction

This pipeline setup is very simple, but totally different from traditional pipeline setup. Because our studio work happens in different manner, ie different peoples are working from different places. So maintain consistency of workflow is very hard. The pipeline tool will take care of consistency of the project. All kind of data transfer from one department to another via Pipeline Tools.

The most important part of it to establish well-defined guidelines, practices, and disciplines, each of our user should well aware of our pipeline tool.

Here, we can look at the Linear workflow. What I mean by linear? Data travel with a specific hierarchy structure.



Pipeline core module has been developed in Python-3.7 64Bit, this module only work in Windows 10 64Bit or upper version.

## 2. Studio Resources

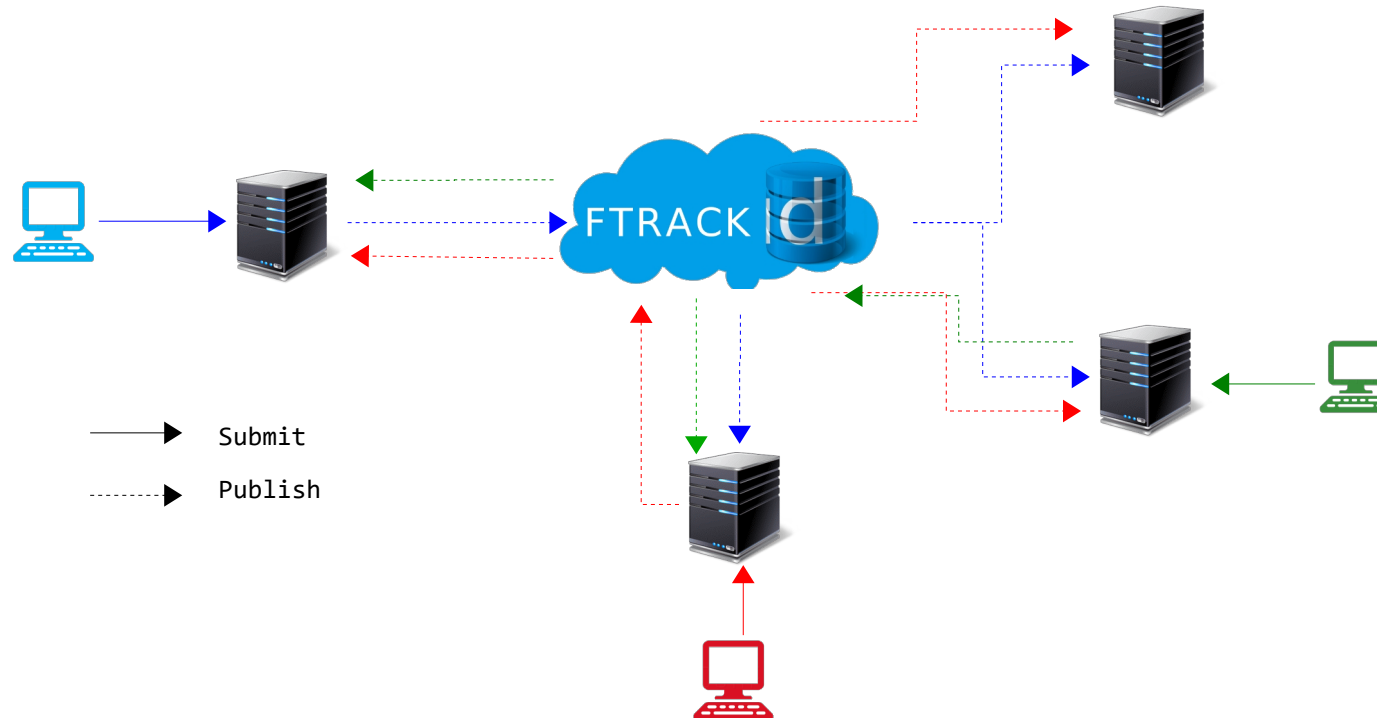
Here, our resources are local storage with different locations and one common **ftrack** storage. Our Pipeline tools will help to maintain this data structure on all local storage. And this should be an automation process.

**Ftrack (Cloud)** is the Global storage unit, this is our final valid data storage unit. And this Global storage unit can sync with each location storage with the help of **Taiko Browser** Tool. For example, Once local workspace submit or publish happened, first it will store local storage and that new data will upload to ftrack storage. If this process happened that can download to any other local storage. The result is identically the same in all local storage.

**Workspace** is the local user PC. For each location workspace, pipeline tools maintain the same discipline, to avoid complexity.

**Storage** is the local location storage unit. All project data and pipeline setups are set up on the local location storage. pipeline tools maintain the same discipline, to avoid complexity. This also same as Workspace, but production need to maintain the submit and publish. Please avoid invalid data in the local storage unit and maintain like a final project data.

*Look at the below flowchart to get more details,*



## 2. User Discipline

We need to create user account for all user, this is very important because we need track user task. Pipeline tool behave based on the user discipline. We have three groups such as guest-user, standard-user and super-user.

and seven user types. The user privileges has to decided based on under the what user group.

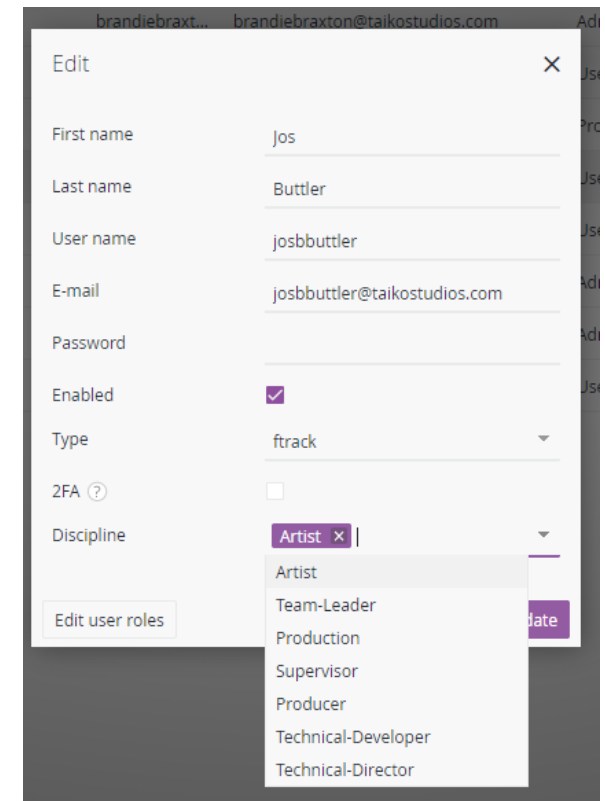
Gust-User - start and submit his tasks

Super-User - Approve, decline and publish the tasks

Standard-User - start, submit, approve, decline and publish his and others tasks

Type	Group
1. Artist	guest-user
2. Team-Leader	standard-user
3. Production	super-user
4. Supervisor	super-user
5. Producer	super-user
6. Technical-Developer	guest-user
7. Technical-Director	super-user

In the Ftrack, there is a new attribute called **discipline**, this is the custom attribute. While create the user, update the value of discipline attribute, this should be **mandatory**, look at the image



The screenshot shows the 'Edit' form for a user in Ftrack. The form includes fields for First name (Jos), Last name (Buttler), User name (josbbuttler), E-mail (josbbuttler@taikostudios.com), Password, Enabled (checked), Type (ftrack), 2FA (unchecked), and Discipline. The Discipline dropdown menu is open, showing a list of roles: Artist, Team-Leader, Production, Supervisor, Producer, Technical-Developer, and Technical-Director. The 'Artist' role is currently selected.

### 3. Project

While create project in ftrack three attributes are mandatory, such as `name`, `full name`, and `thumbnail`.

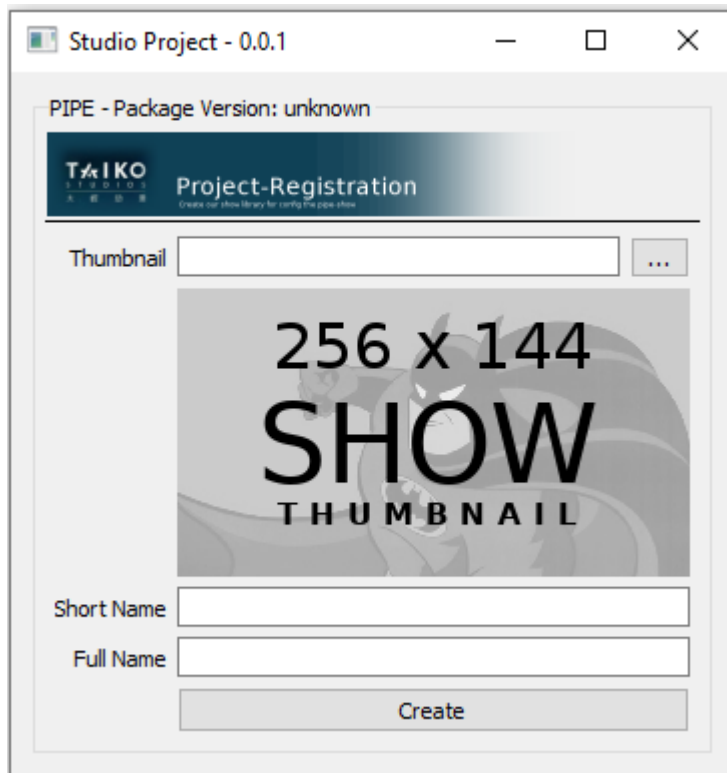
For example

```
name = "MSH"
```

```
full name = "my super hero"
```




```
thumbnail = "your local directory, format should be *.png or *.jpg and resolution 256x144"
```

Pipeline tool will help to generate the project. But `super-user` have only the permission to create the project.



## 4. Category

Under the project we are divided in to two category, such as Assets and Scene, so the term category means assets and scene. Once you create the project categories are generate automatically.

Tasks			Type
Count (task) 24			
1		▼ 🏠 <a href="#">My Super Hero</a>	Project
2		➤ 📁 <a href="#">assets</a>	Folder
3		➤ 📁 <a href="#">scene</a>	Folder

## 5. Step or Task

Under the category we are divided in to different steps, all steps are the tasks. Our pipeline should support to add more steps. Create tasks using use pipe tool **mandatory**, because all dependency task will generate automatically. Once you create task then you can go to ftrack assign the task to respective users, also you can do your on changes All the task assignment has to done via ftrack. There is no tool for task assignment.

Studio Tasks - 0.0.1

PIPE - Package Version: unknown

TAIKO

TASK-Registration

Create our task library for coding the pipeline

Project

My Super Hero

Category

assets


Asset Name

Tier

Camera

Thumbnail

1024 x 1024



Description

Create

Art	Assets
Model	Assets
Lookdev	Assets
Groom	Assets
Puppet	Assets
Layout	Scene
Animation	Scene
Render	Scene
Compositing	Scene

assets	Folder
Batman	Asset Build (Charac...
art	Task (art)
model	Task (model)
lookd...	Task (lookdev)
groom	Task (groom)
puppet	Task (puppet)



scene	Folder
101	Sequence
1001	Shot
render	Task (render)
animation	Task (animation)
layout	Task (layout)
compositing	Task (compositing)

## 6. Versions

We have two kind of versions such as **submit** and **publish**, the term we can call **kind**. And we are following Semantic Versioning, ie version number MAJOR.MINOR.PATCH, increment. More details <https://semver.org/>.

Submit and publish are its on versions, for example submitted version 0.2.0 is approved so publish version should be 0.0.1 or increment of publish versions

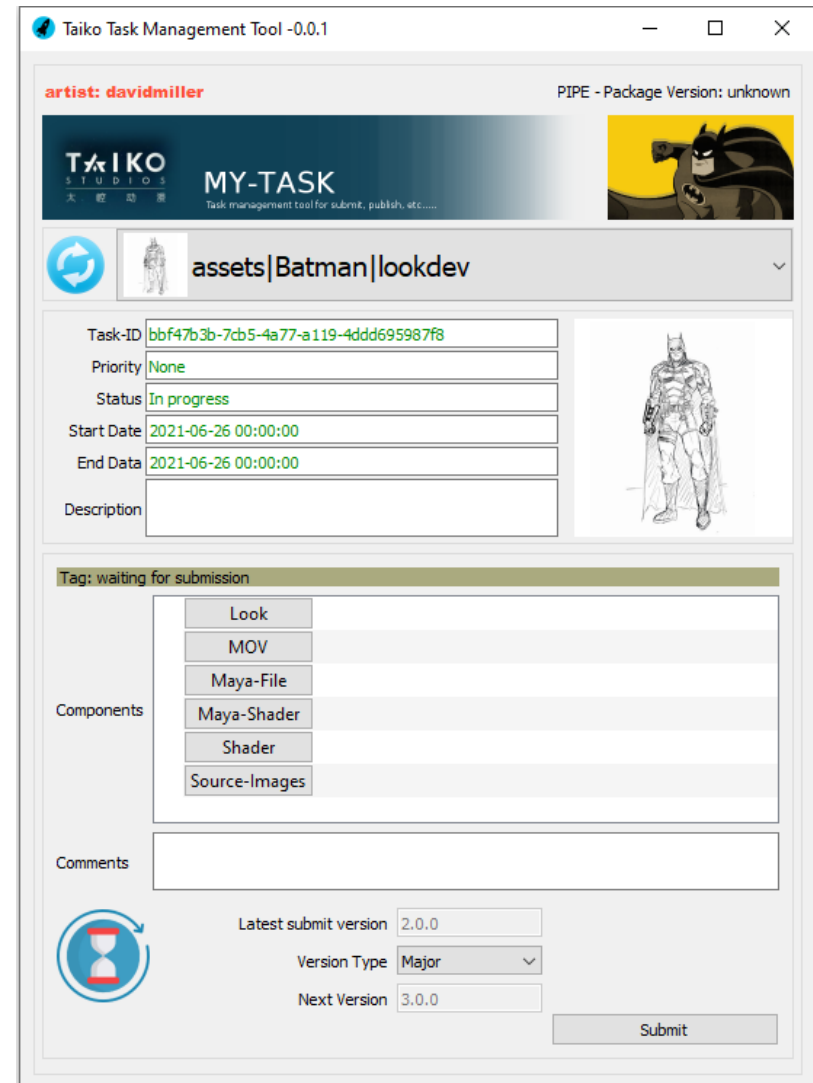
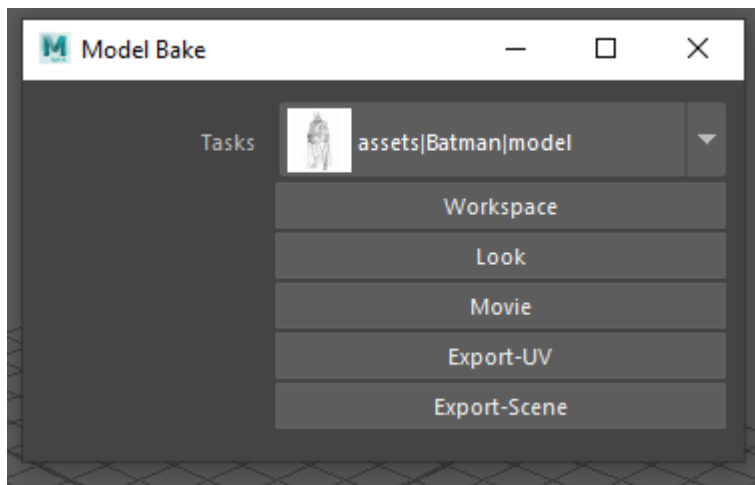
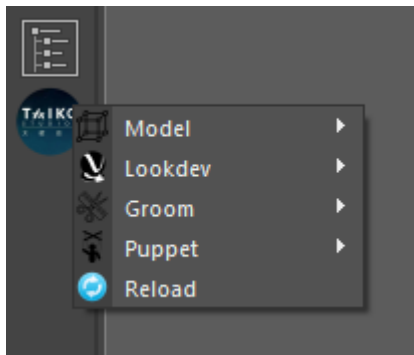
1. **Submit** - has to done by task assignee. Once task done he/she can submit the task.
2. **Publish** - has to done by task assigner (super-user or standard-user).

		Version Link	Taiko-Version	Status	Asset type	Asset name	Published by
1		<a href="#">assets / Batman / assets Batman puppet v1</a>	0.0.1	Comple...	publish	assets Batman puppet	Shaofu Zhang
2		<a href="#">assets / Batman / assets Batman puppet v1</a>	0.0.1	Approved	submit	assets Batman puppet	David Miller
3		<a href="#">assets / Batman / assets Batman model v6</a>	4.0.0	Comple...	publish	assets Batman model	Shaofu Zhang
4		<a href="#">assets / Batman / assets Batman model v16</a>	2.0.0	Approved	submit	assets Batman model	Jos Buttler
5		<a href="#">assets / Batman / assets Batman art v12</a>	7.0.0	Comple...	publish	assets Batman art	Shaofu Zhang
6		<a href="#">assets / Batman / assets Batman art v7</a>	5.0.0	Approved	submit	assets Batman art	David Miller



## 7. Version Components

Versions contains components(each task has its on components). Tool will help to extract the components from the maya. That tools I term as a Bake such as Model Bake, Lookdev Bake, Puppet Bake, etc. Once extract the components, use My Task tool to submit the components with in increment version

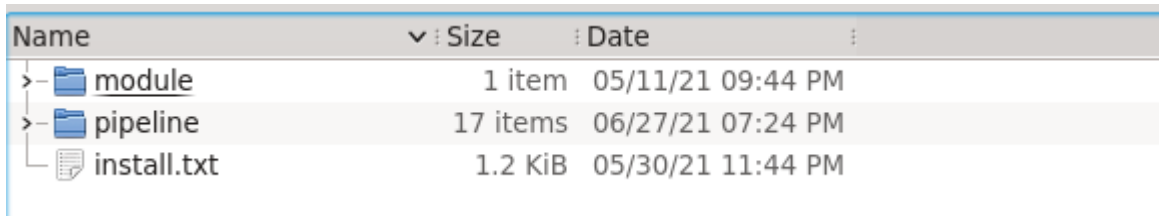


## 8. Configure the Pipeline Package and Project

> **In-house**, no need to worry about configuration of the pipeline package and project. Let's start from Start the Project page

> **Freelancer or any one work from his personal computer**, just follow the below instruction.

1. Download the Pipeline Package from respective link, we will send the package link
2. Extract the contents of the package to your local directory, user can choose the directory, I recommend My-Document directory is the best practices.
3. Two folder such as module and pipeline.



Name	Size	Date
module	1 item	05/11/21 09:44 PM
pipeline	17 items	06/27/21 07:24 PM
install.txt	1.2 KiB	05/30/21 11:44 PM

4. Open the TAIKO.bat file inside the package and edit the PROJECT-DIRNAME (line 20) to your local directory. And edit the **USER-TYPE=** to **freelancer**

*for example*

*set PROJECT-DIRNAME=your project directory*

*set PROJECT-DIRNAME=D:/taiko-studio/projects*

*set PROJECT-DIRNAME=D:/test/works/projects*

*please not does not add your project folder name*

*set USER-TYPE=freelancer*

Name	Size	Date
> module	1 item	05/11/21 09:44 PM
> pipeline	17 items	06/27/21 07:24 PM
> animation	1 item	06/27/21 07:00 PM
> apis	1 item	06/27/21 07:00 PM
> compositing	1 item	06/27/21 07:00 PM
> example	3 items	06/27/21 07:00 PM
> groom	1 item	06/27/21 07:00 PM
> lookdev	1 item	06/27/21 07:00 PM
> modeling	1 item	06/27/21 07:00 PM
> pipe	1 item	06/27/21 07:00 PM
> puppet	1 item	06/27/21 07:00 PM
> rendering	1 item	06/27/21 07:00 PM
> resources	2 items	06/27/21 07:00 PM
> startup	1 item	06/27/21 07:00 PM
> toolkits	1 item	06/27/21 07:00 PM
config.json	1.1 KiB	06/25/21 12:51 PM
README.md	179 B	05/12/21 01:34 PM
setup.py	4.6 KiB	06/21/21 05:50 PM
TAIKO.bat	1.3 KiB	06/21/21 04:59 PM
install.txt	1.2 KiB	05/30/21 11:44 PM

```

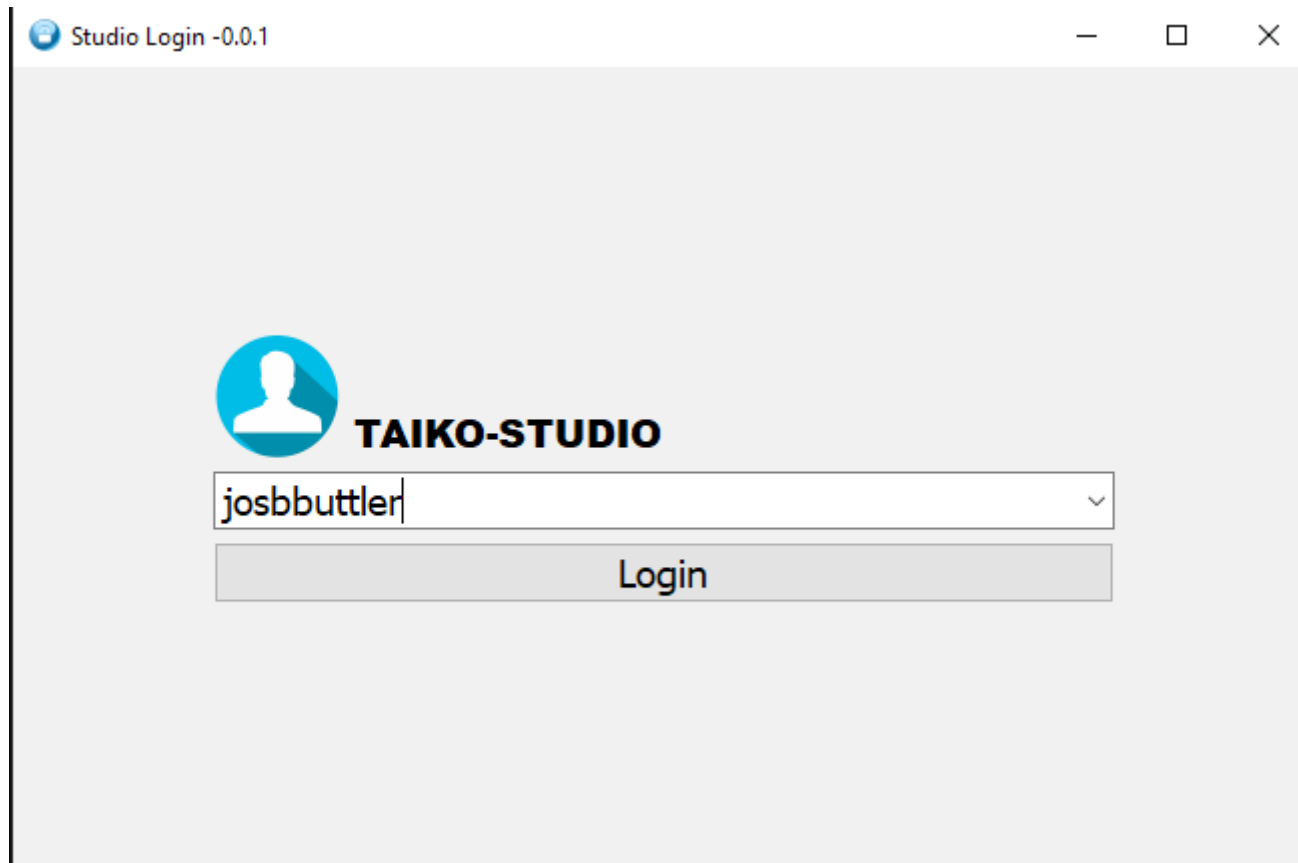
18
19 : update the PROJECT-DIRNAME, PYTHON-EXE as per your requerments
20 set PROJECT-DIRNAME=B:/shows
21 set USER-TYPE=in-house or freelancer
22

```

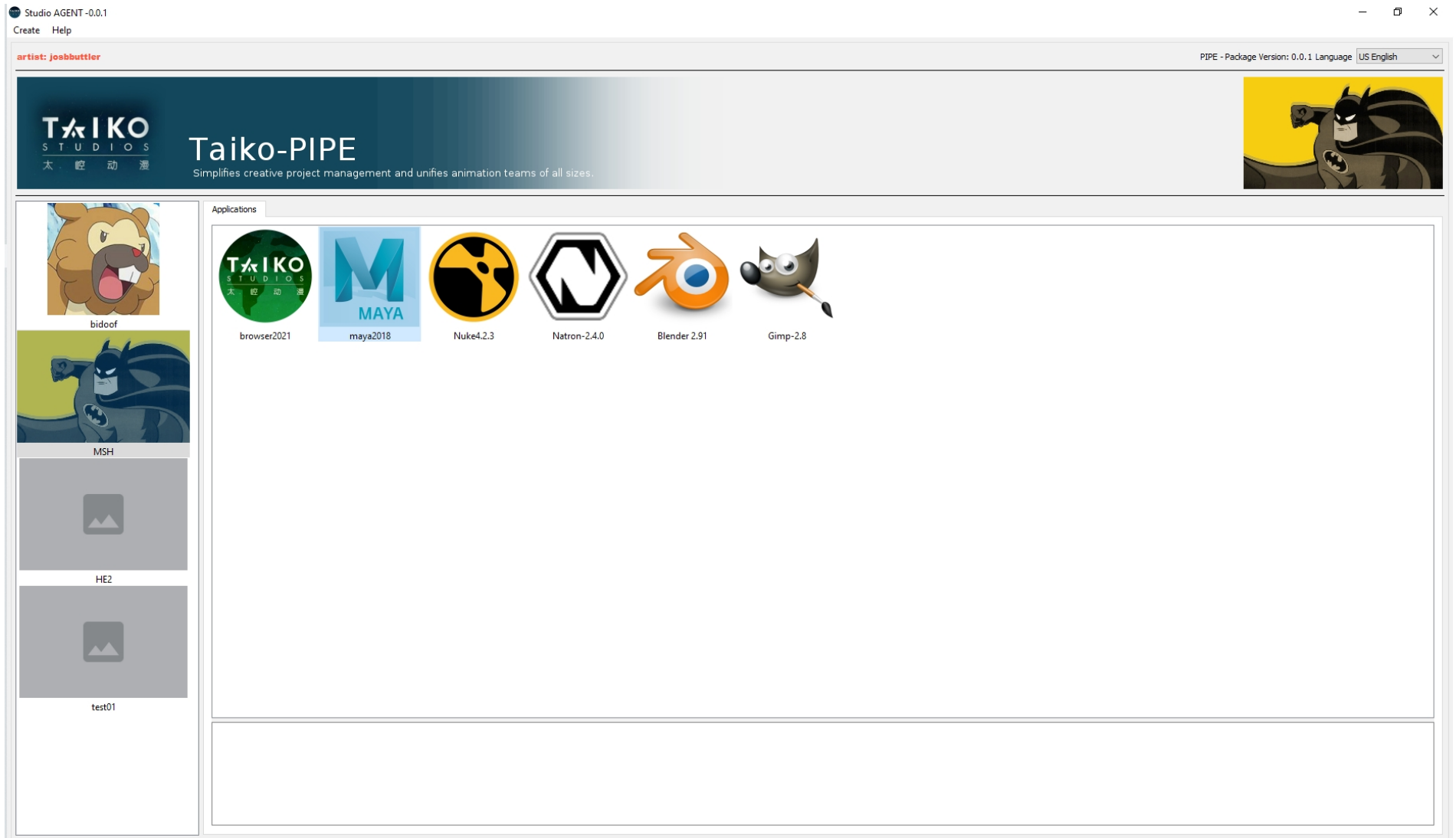
4. Open the Command Prompt drag and drop the TAIKO.bat and press enter or directly double click on the TAIKO.bat file

## 9. Let's start the works

> **Step 1**, login with your ftrack user name



> **Step 2**, After login you will get the **studio agent** tool, this is core tool for all user. All user need to use this tool to launch all application like maya, nuke, etc.



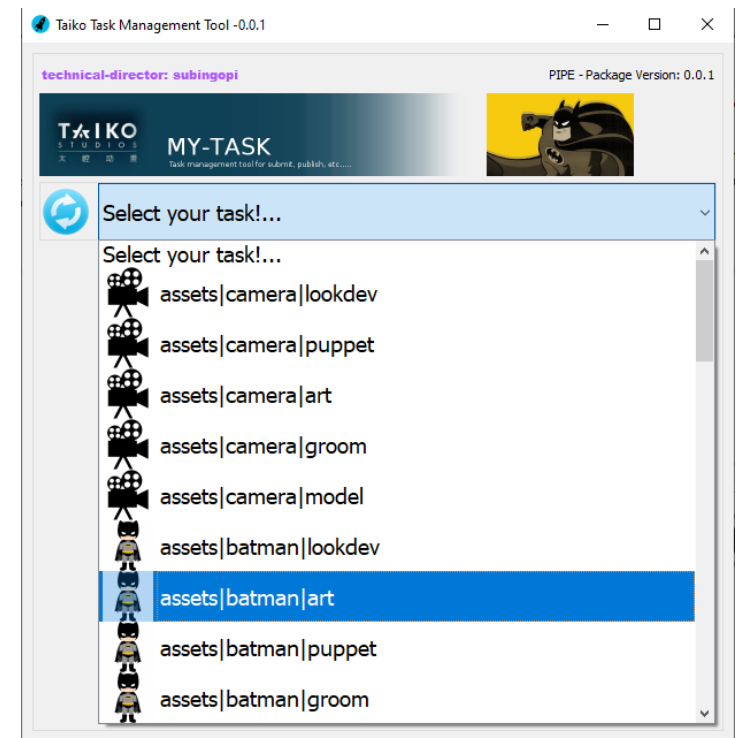
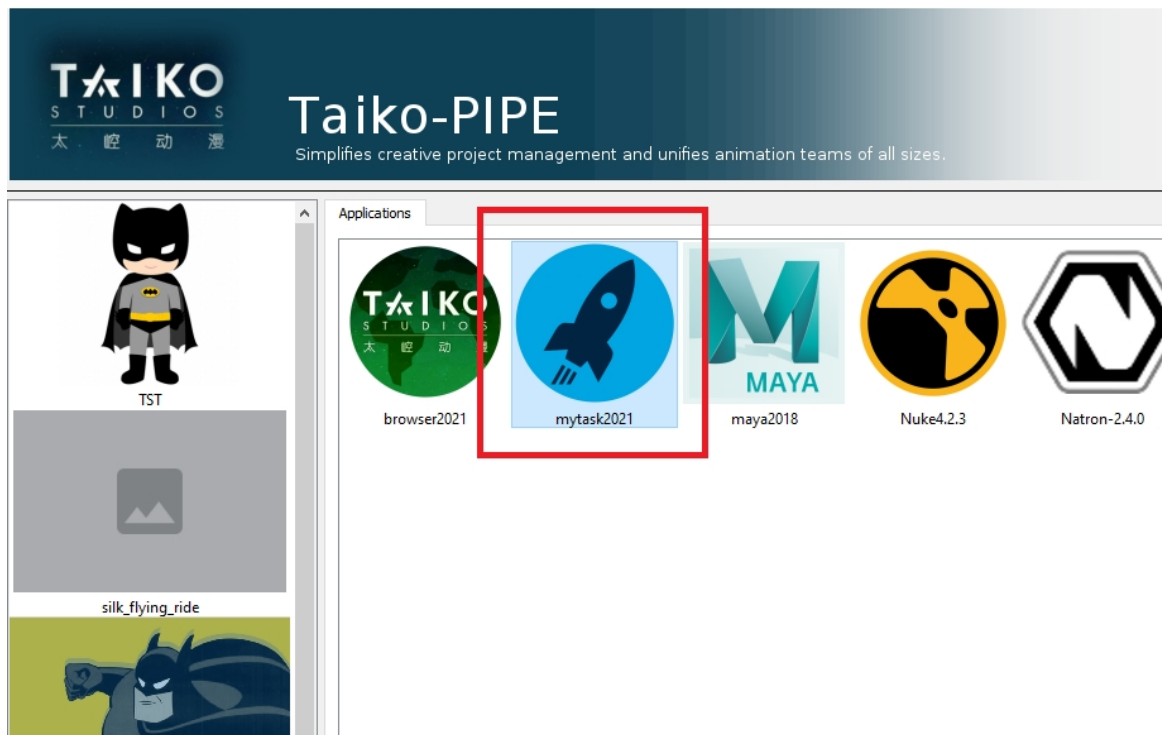
## How to configure Studio Agent Tool ?

*pipeline\pipe\0.0.1\pipe\resources\inputs\applications.json*, update the path of your software, in below maya is the one of the example, if you want to add new software, just duplicate the maya and rename to as per your requirements

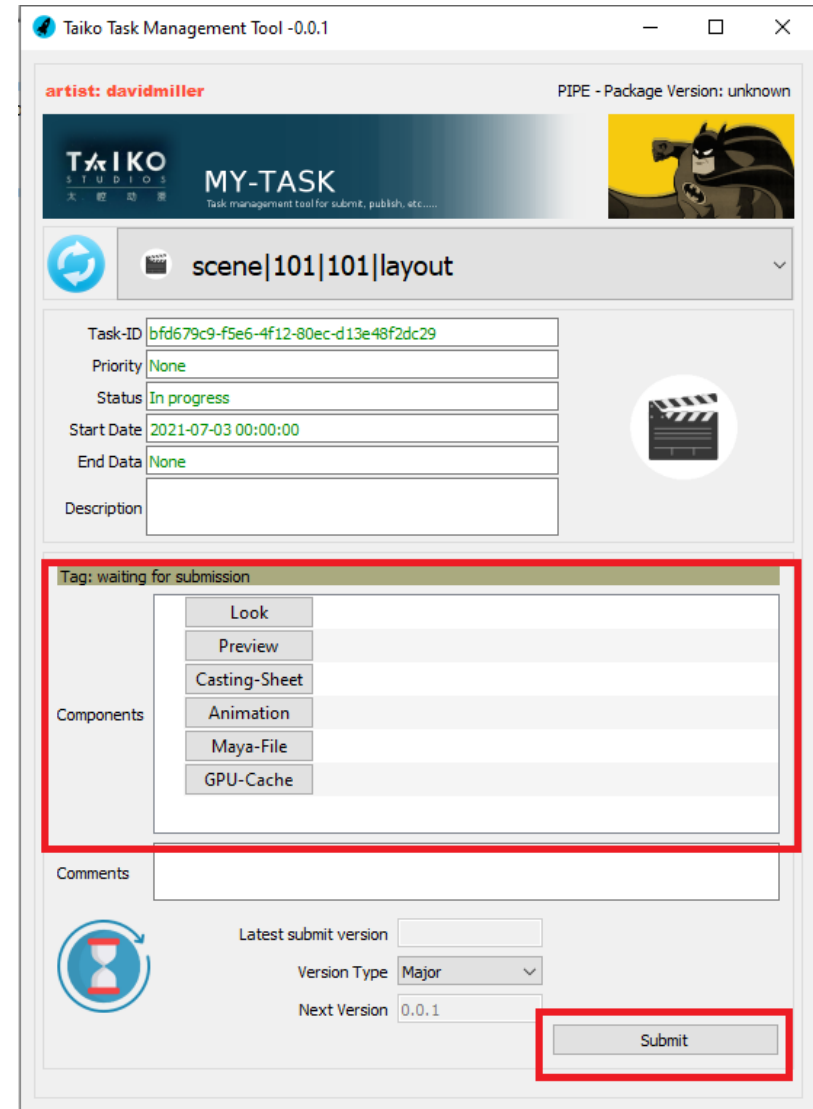
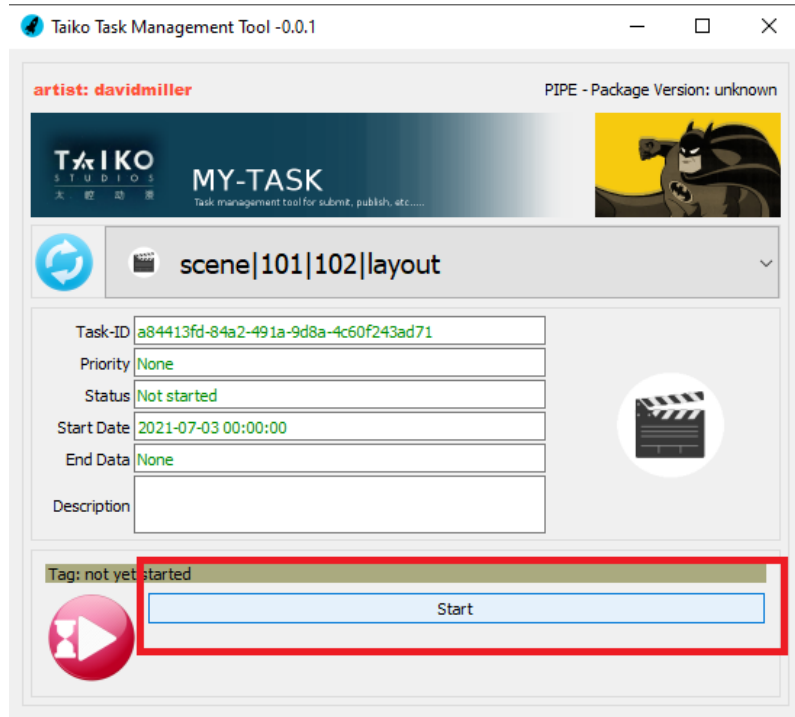
```
{
  "order": 3,
  "name": "maya2018",
  "global": "maya",
  "production": false,
  "enable": true,
  "version": {
    "value": "maya2018",
    "env": "MAYA_VERSION",
    "enable": true
  },
  "path": {
    "value": "C:/Program Files/Autodesk/Maya2018",
    "env": "MAYA_PATH",
    "enable": true
  },
  "exe": {
    "value": "C:/Program Files/Autodesk/Maya2018/bin/maya.exe",
    "env": "MAYA_EXE",
    "enable": true
  },
  "icon": {
    "value": "maya2018.png",
    "env": "MAYA_ICON",
    "enable": true
  },
  "language": {
    "value": "en_US",
    "env": "MAYA_UI_LANGUAGE",
    "enable": true,
    "values": [
      "zh_CN",
      "en_US",
      "ja_JP"
    ]
  }
},
```

> **Step 3**, Select your project from right column, double click on the maya icon from second column

> **Step 4**, Select your project from right column, double click on the mytask2021. My-Task Tool will drive your tasks. And user can see his/her tasks, to load takes time, please wait until open the tool, more details, just look at the Command Prompt console.



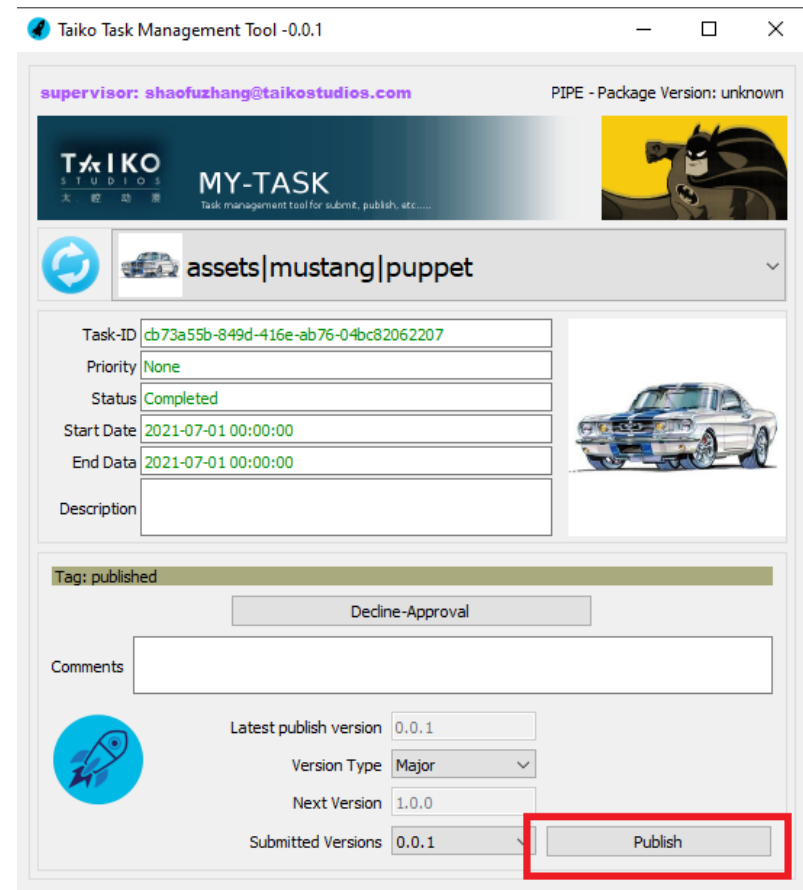
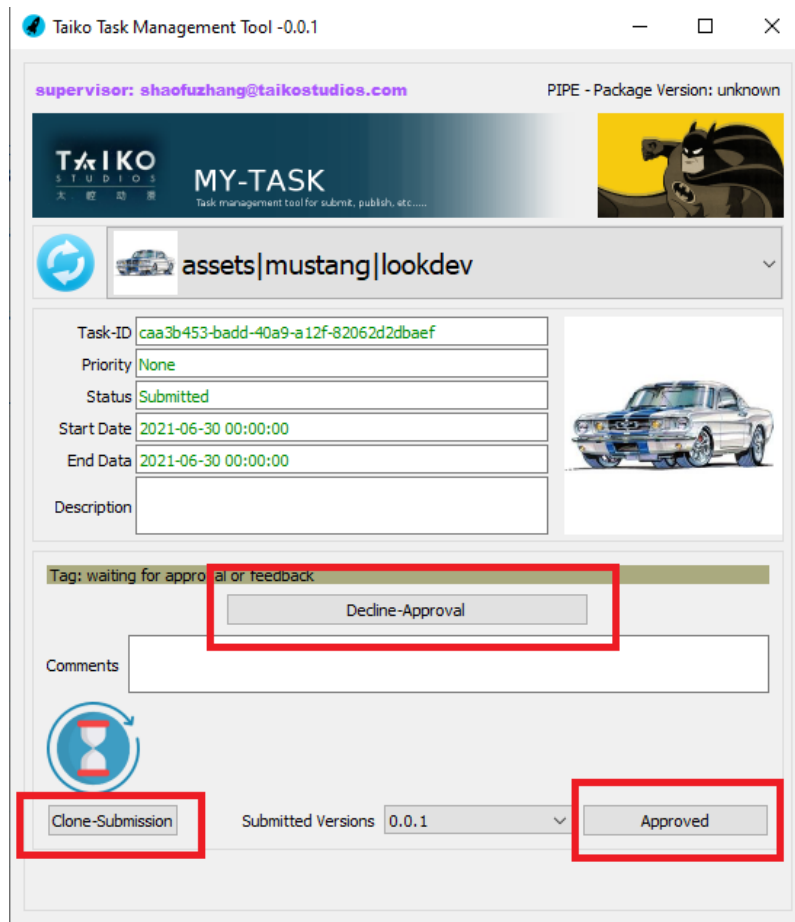
> **Step 4**, (This is normal user level) Select your task and click on start button, once its done submit his/her task.  
(Submit components use the bake tools from maya.)





> **Step 5**, This level only for super users, they can approve or decline the task. Super user can see all tasks. Click on Clone-Submission button to download the user submission, and super user can review the task

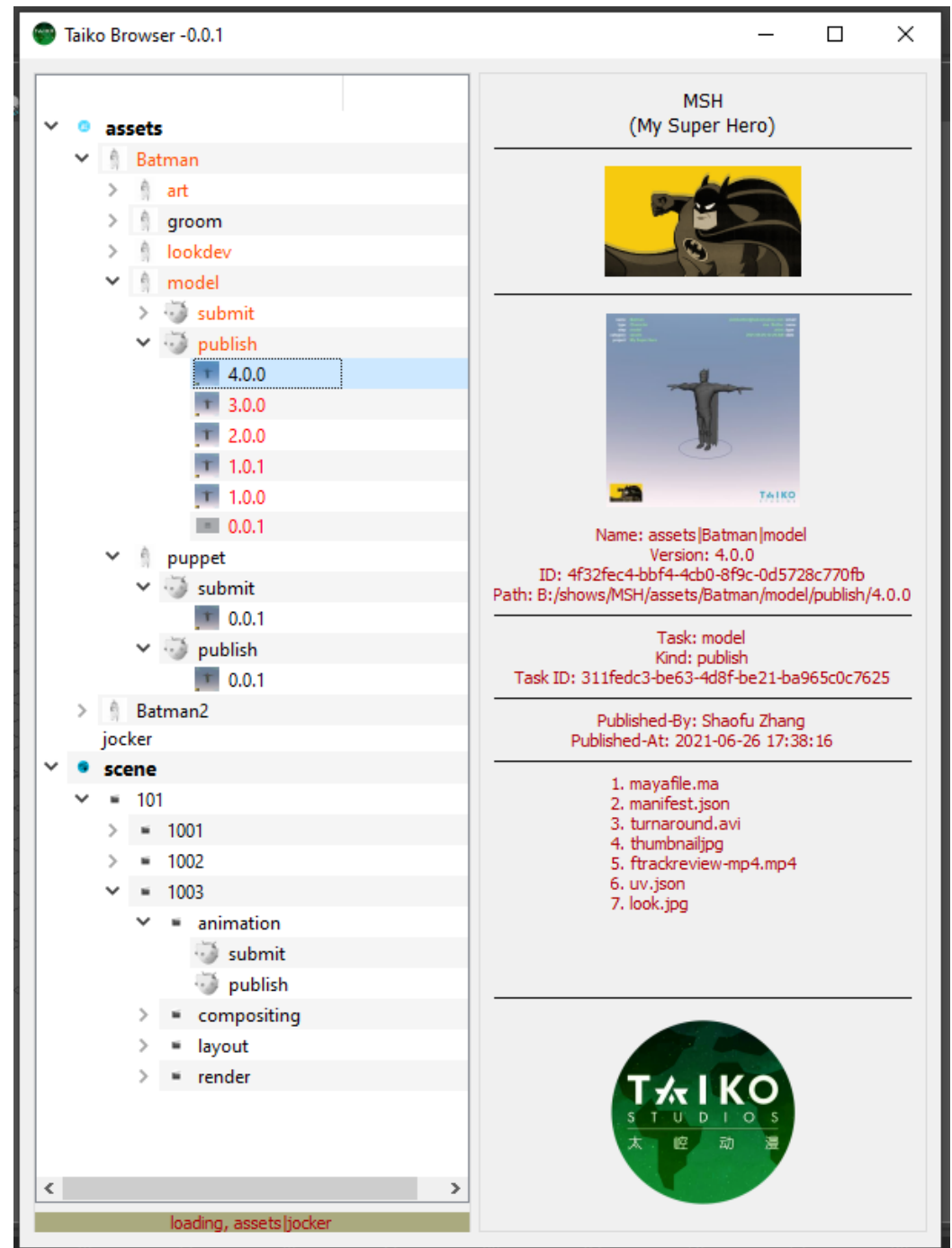
> **Step 6**, Then click on the publish button



> **Step 7, Taiko Browser**, this the task browser tool, all user can use this tool to get the status and download the task and its components. Use this tool to generate the assets and shots. This will make and maintain the well folder structure. Even if you removed any components, user can re-generate all elements in our show.

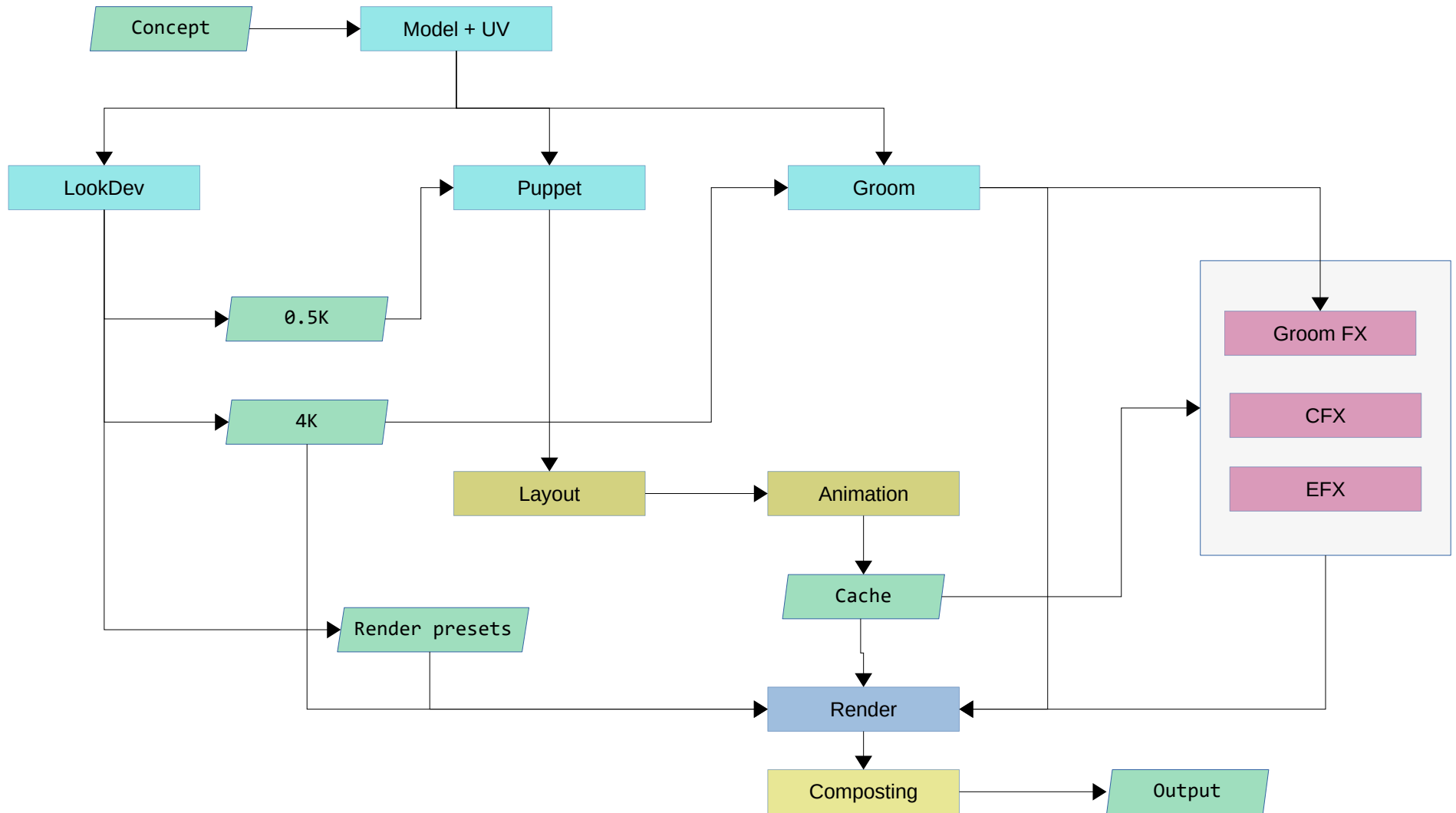
User can access this tool from maya as well.

To open, import and reference the asset and scene. Use only this tool, then only pipeline can track all **scene level** submission and publish. In Asset level if you want to download the dependency, use this tool to download the respective dependency step or task, by default once user click on start button from My-Task automatically download the dependency steps or tasks.



## 10. Work flow

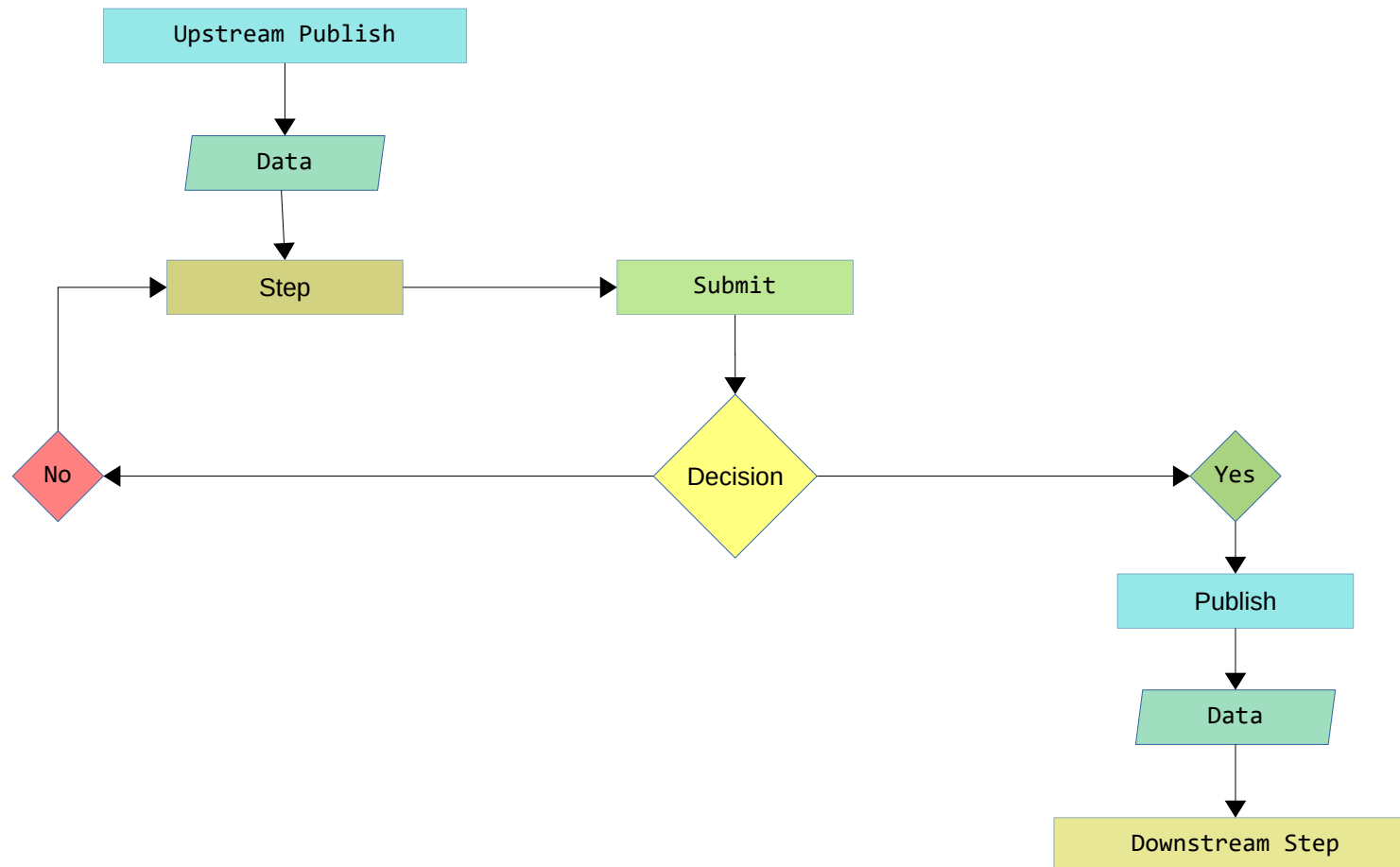
Project Workflow: this is our data transfer flow. Just go with the below workflow.



## 11. Detailed Work flow with specific step

Here look at the below flow chart, data is coming from upstream step publish, and once it comes to the current step user need to submit his/her file, If it is not approved its go back to the user with messages, and he/she fix the bug and need to submit again, then is it approved, it goes to publish process. And publish data has to block (no modification). If you want to edit the existing publish, he/she can publish as a next version.

The advantage of submitting, we can avoid multiple publish processes. Submitted data are temporary files, once it's approved we can delete it. Either each correction we need to make each publish. I recommend this publish data should be valid and should pass to downstream steps. For the review proposes, we can use submit process instead of publishing. This one of the best practices to optimize our storage.



## 12. Folder Structure

Is it never matter what kind of folder structure we are following, because all valid data we handle through the pipeline tools, but pretty and standard folder structures avoid much confusion and easy to find the data. I propose the below folder structure. This is the simple, efficient, and standard folder structure. The editorials I am skipping because I would like to discuss more with supervisor.

```
└─ show1
   └─ editorials
   └─ assets
   └─ scene
      └─ layout
      └─ animation
      └─ render
      └─ composting
```

```
└─ boy
   └─ model
      └─ publish
         └─ 0.0.1
         └─ 0.0.2
      └─ submit
         └─ 0.0.1
         └─ 0.0.2
         └─ 1.0.0
      └─ works
   └─ lookdev
      └─ publish
         └─ 0.0.1
      └─ submit
         └─ 0.0.1
      └─ works
   └─ puppet
      └─ publish
         └─ 0.0.1
      └─ submit
         └─ 0.0.1
      └─ works
```

```
└─ animation
   └─ sequence_101
      └─ shot_101
         └─ publish
            └─ 0.0.1
            └─ 0.0.2
         └─ submit
            └─ 0.0.1
            └─ 1.0.0
         └─ works
└─ composting
   └─ sequence_101
      └─ shot_101
         └─ publish
            └─ 0.0.1
         └─ submit
            └─ 0.0.1
            └─ 1.0.0
         └─ works
└─ layout
   └─ sequence_101
      └─ shot_101
         └─ publish
            └─ 0.0.2
         └─ submit
            └─ 1.0.0
         └─ works
└─ render
   └─ sequence_101
      └─ shot_101
         └─ publish
            └─ 0.0.1
         └─ submit
            └─ 0.0.1
            └─ 0.0.2
            └─ 1.0.0
         └─ works
```

## 13. Versioning

For versions, I prefer Semantic Versioning.

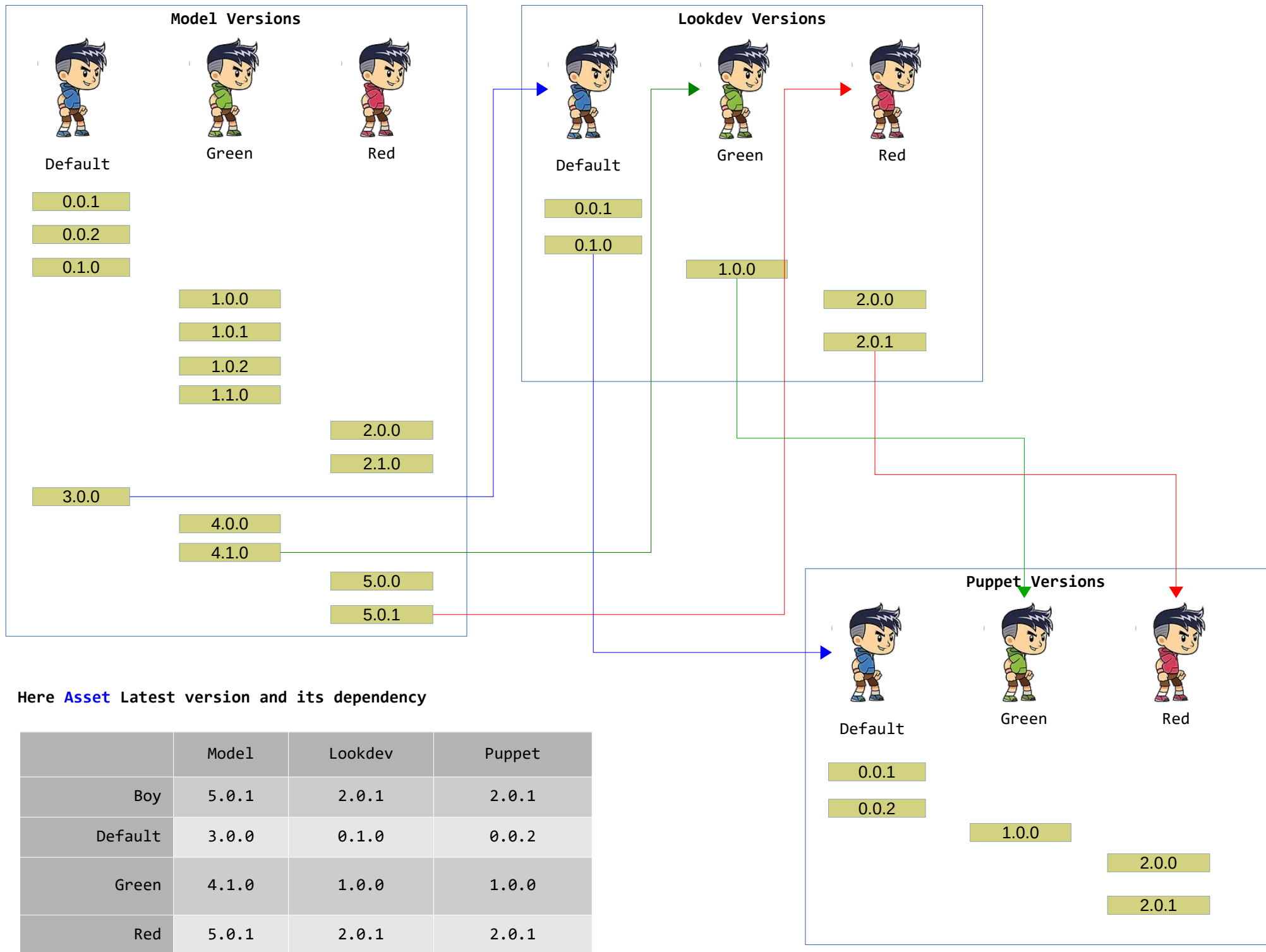
- MAJOR - when you make incompatible changes.
- MINOR - when you add functionality in a backward-compatible manner
- PATCH - when you make backward-compatible bug fixes.

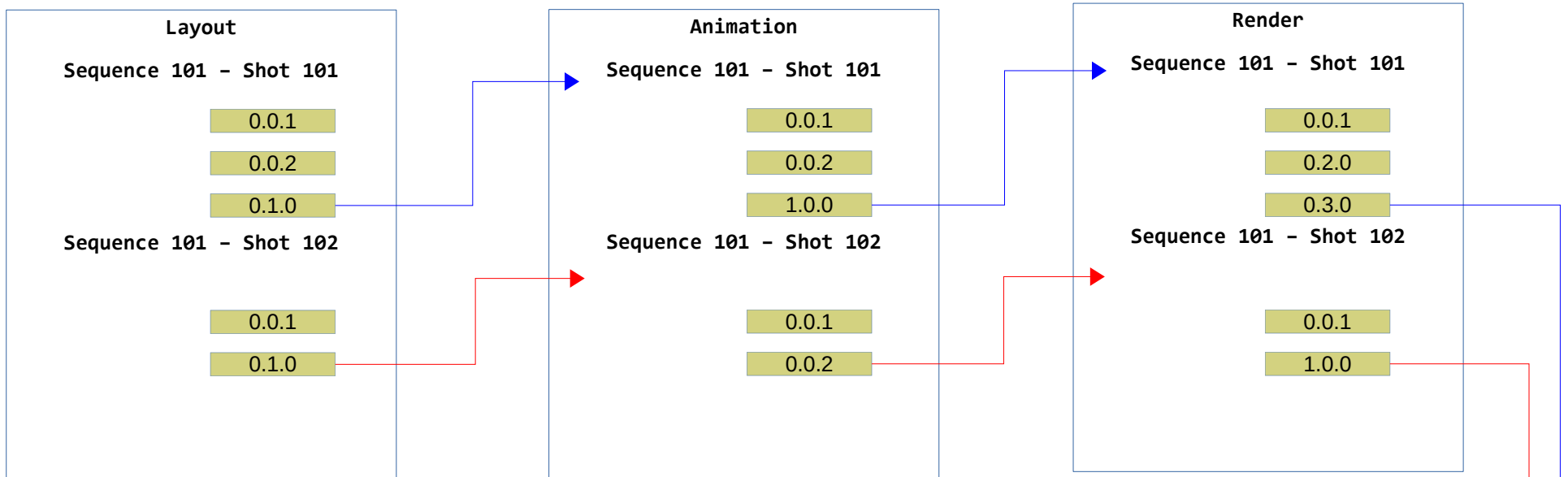
More details <https://semver.org/>

Here variants and versions we can include with semantic versions, Instead of v1, v2, v3, this is very efficient and visually understand what kind of changes happened, is it major or minor or patch.

And any kind of versions and dependency conflict happened our tool will find out. Most probably conflict never happened because publish and build process done by Pipeline tools. And we have a version management tool as well.

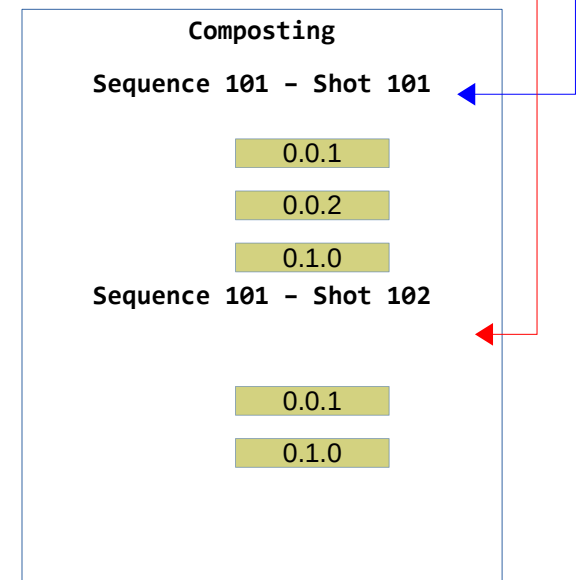
Look at the below flowchart to get more ideas.





Here **Shot** Latest version and its dependency

	Layout	Animation	Render	Composting
101/101	0.1.0	1.0.0	0.3.0	0.1.0
101/102	0.1.0	0.0.2	1.0.0	0.1.0





## 14. Step Published Data Types

Here, the output data might be change as per our requirement, I add some basic output data for the respective steps.

Steps	Outputs
Model + UV	Model Scene, UV Data, Alembic Cache, Scene Description
LookDev	Lookdev Scene, 4k and 0.5k Source Images, Render Presets, Scene Description
Puppet	Puppet Scene, Scene Description
Groom	Groom Scene, Scene Description
Layout	Layout Scene, Animation Data, Asset Scene Description, Movie, Scene Description
Animation	Animation Scene, Animation Data, Asset Scene Description, Animation Cache, Movie, Scene Description
Render	Render Scene, Asset Scene Description, Render outputs, Movie, Scene Description
Groom FX	Groom Scene, Groom Cache, Movie, Scene Description
FX	EFX Scene, EFX Cache, Movie, Scene Description
Composting	Composting Scene, Output movie, Scene Description

## 15. Core Pipeline Production Tools

- **Studio Agent**  
To configure and launch applications and software.
- **My-Task**  
Task Management tool.
- **Taiko Browser**  
Get the status and download the task and its components.
- **Maya Steps Bake Tools**  
Extract the components from the maya for submit the step tasks.

## 11. Pipeline Resources

- Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
- Python 2.7.18 (v2.7.18:8d21aa21f2, Apr 20 2020, 13:25:05) [MSC v.1500 64 bit (AMD64)] on win32
- PySide
- FTrack API
- PILLOW
- opencv-python 4.5.2.54

## 16. Pipeline Frameworks and Packages (Repository)

To address inconsistencies and duplicated development efforts, in our case to develop a production pipeline. Each framework has its classes contained by packages. All frameworks are each repository and frameworks are no dependent on any other repositories. Frameworks are standalone packages. And all repositories are dependent on the framework but and no dependency on other repositories.

which you are going to see the examples

1. animation
2. apis
3. compositing
4. groom
5. lookdev
6. modeling
7. pipe
8. puppet
9. rendering
10. resources
11. startup
12. toolkits